9. 반복문

주요 내용

- `while` 반복문

9.1. 변수 재할당/업데이트

변수 재할당

In [1]:

```
x = 5
y = x
x = 7
```

In [2]:

```
Z = x + y
print(Z)
```

```
12
```

변수 업데이트

In [3]:
```
x = x + 1
```

In [4]:
```
x = x - 1
```

변수 간편 업데이트

활용	의미
`x += 1`	`x = x + 1`
`x -= 1`	`x = x - 1`
`x *= 2`	`x = 2*x`
`x /= 2`	`x = x / 2`
`x **= 2`	`x = x**2`
`x //= 2`	`x = x //2`
`x %= 2`	`x = x % 2`

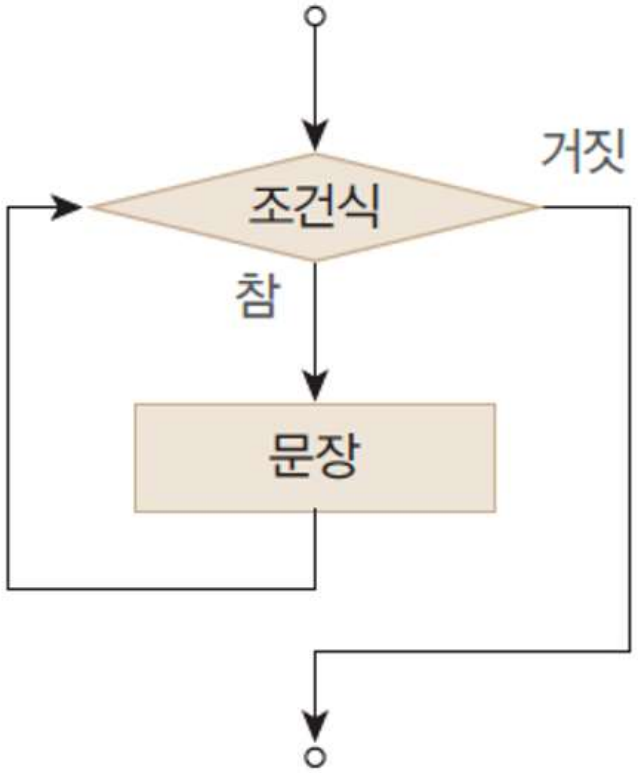
In [5]:

```
x = 5
x /= 2
print(x)
```

```
2.5
```

9.2. `while` 반복문

반복문 실행 단계



In [6]:

```
def countdown(n):
    while n > 0:     # n 이 0 보다 큰 경우
        print(n)
        n = n - 1

    print('발사!')    # 기저 조건: n 이 0 인 경우
```

In [7]:

```
countdown(3)
```

```
3
2
1
발사!
```

In [8]:

```
countdown(5)
```

```
5
4
3
2
1
발사!
```

for 반복문 활용

In [23]:

```
def countdown_for(n):
    for n in range(n, 0, -1):
        print(n)

    print('발사!')
```

In [24]:

```
countdown_for(3)
```

```
3
2
1
발사!
```

9.3. 무한 루프

예제

- 아래 함수를 양의 정수와 함께 호출하면 `while` 반복문이 무한 반복된다.

In [9]:

```
def count_infinitely(n):
    while n > 0:
        print(n)
        n = n + 1
```

```
>>> count_infinitely(1)
1
2
3
4
...
```

콜라츠 추측

- 무한 루프가 발생할지 여부를 미리 알 수 없는 경우 존재

In [10]:

```
def collatz(n):
    while n != 1:
        print(n, end=" -> ")
        if n%2 == 0:          # 짝수인 경우
            n = n//2
        else:                 # 홀수인 경우
            n = n*3 + 1

    print(1)                  # 기저 조건
```

In [11]:

```
collatz(3)
```

```
3 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1
```

In [12]:

```
collatz(7)
```

```
7 -> 22 -> 11 -> 34 -> 17 -> 52 -> 26 -> 13 -> 40 -> 20 -> 10 -> 5
-> 16 -> 8 -> 4 -> 2 -> 1
```

- `collatz(n)` 을 실행했을 때 언제 실행이 멈출지 알 수 없음

In [13]:

```
collatz(111)
```

```
111 -> 334 -> 167 -> 502 -> 251 -> 754 -> 377 -> 1132 -> 566 -> 28
3 -> 850 -> 425 -> 1276 -> 638 -> 319 -> 958 -> 479 -> 1438 -> 719
-> 2158 -> 1079 -> 3238 -> 1619 -> 4858 -> 2429 -> 7288 -> 3644 ->
1822 -> 911 -> 2734 -> 1367 -> 4102 -> 2051 -> 6154 -> 3077 -> 923
2 -> 4616 -> 2308 -> 1154 -> 577 -> 1732 -> 866 -> 433 -> 1300 ->
650 -> 325 -> 976 -> 488 -> 244 -> 122 -> 61 -> 184 -> 92 -> 46 ->
23 -> 70 -> 35 -> 106 -> 53 -> 160 -> 80 -> 40 -> 20 -> 10 -> 5 ->
16 -> 8 -> 4 -> 2 -> 1
```

무한루프와 `break` 명령문

In [14]:

```
while True:
    line = input('단어입력: ')
    if line == '중지':
        break
    print(line)

print('종료합니다!')
```

```
단어입력: 좋아요
좋아요
단어입력: 파이썬
파이썬
단어입력: 중지
종료합니다!
```

9.4. 무한 루프 활용

게임 프로그래밍

- 전형적인 무한 루프 활용

```
while True:
    # Check for events.
    for event in pygame.event.get():
        if event.type == QUIT:  # ESC 키 누르기
            pygame.quit()
            sys.exit()
        if event.type == KEYDOWN:
            # Change the keyboard variables.
            if event.key == K_LEFT or event.key ==
K_a:
                moveRight = False
                moveLeft = True
                ...
```

- ESC 키를 누를 때 게임을 종료시키는 조건

```
if event.type == QUIT:  # ESC 키 누르기
    pygame.quit()
    sys.exit()
```

제곱근 계산: 뉴튼 방법

- a 의 제곱근 $\sqrt{a}$을 구하는 점화식

$$x_0 = a$$

$$x_{n+1} = \frac{x_n + \frac{a}{x_n}}{2}$$

- 점화식을 파이썬에서 변수 업데이트로 구현

```
x = a
x = (x + a/x) / 2
```

$\sqrt{2}$ 계산

- numpy 모듈의 `sqrt()` 함수 활용 가능

In [15]:

```
import numpy as np

np.sqrt(2)
```

Out[15]:

```
1.4142135623730951
```

$\sqrt{2}$ 계산: 뉴튼 방법

In [16]:
```
a = 2
x = a
```

In [17]:
```
x = (x + a/x) / 2
x
```

Out[17]:
```
1.5
```

In [18]:
```
x = (x + a/x) / 2
x
```

Out[18]:
```
1.4166666666666665
```

In [19]:
```
x = (x + a/x) / 2
x
```

Out[19]:
```
1.4142156862745097
```

무한 반복문 활용

- 반복문 활용 가능. 단, 무한 반복이 이뤄지도록 해야 함.
- 이유는 언제 끝날지 모르기 때문

```
a = 2
x = a
while True:
    x = (x + a/x) / 2
```

오차 허용 한도

- 무한 루프가 발생하지 않도록 오차 허용 안에 들어오면 실행을 멈추도록 하는 장치

In [20]:

```
def my_sqrt(a, epsilon):
    x = a
    while True:
        y = (x + a/x) / 2
        if abs(x - y) < epsilon:    # 오차 범위 안이면 실행 중지
            break
        x = y                       # x 업데이트
    return y
```

In [21]:

```
my_sqrt(2, 0.0001)
```

Out[21]:

```
1.4142135623746899
```

In [22]:

```
my_sqrt(2, 0.000001)
```

Out[22]:

```
1.4142135623730951
```